FROM: Noah Kunin
Infrastructure Director, 18F | GSA

SUBJECT: DNSSEC Compliance

Executive Summary

18F is notifying interested parties it is unable to comply with DNSSEC due to flaws in the underlying standard. 18F has implemented a **HTTPS Only Standard** as a workaround to DNSSEC gaps and as a fundamental prerequisite to true end-to-end security over untrusted networks. 18F has also worked with OMB to implement a HTTPS Only policy for the entirety of the Federal government.

DNSSEC cannot be *truly* implemented per OMB guidance due to:

- No web browsers provide a user interface or programmatic indicator that DNSSEC has failed or succeeded
- Multiple gaps in the chain of trust
- The Authenticated Data (AD) bit of DNS packets, indicating a DNSSEC lookup has succeeded, can be easily forged
- SHA-1, the most common encryption method used with DNSSEC can be cracked, and has been formally *disallowed* by NIST

As a result:

- 18F and GSA continue to **create a trusted communication channel** from the DNS root to second-level domains (ex: cloud.gov) via its possession and technical control over dotgov.gov and role as the underlying registrar for all of ".gov".

- 18F ensures all systems being served from its infrastructure and platforms **conform to the standards set forth in the Federal HTTPS Only Standard**. This ensures that if DNS is poisoned during any communication, an error will be returned to any end-user using a web browser as their client.

- 18F recommends that all Federal agencies **do not extend trust on the basis of DNS.** Instead, agencies should only use **multi-factor authentication** in order to transmit sensitive data.

Policy Background

In August 2008, then Administrator of the Office of E-Government and Information Technology, Karen Evans, signed and circulated OMB Memo M-08-23 “Securing the Federal Government’s Domain Name System Infrastructure”. The memo articulated new policy requiring all agencies to deploy “appropriate DNSSEC capabilities” to all “applicable information systems” by December 2009.

The memo was released one month after US-CERT validated the “Kaminsky DNS vulnerability” also known as “DNS Insufficient Socket Entropy Vulnerability” or CVE-2008-1447. This flaw showed that *recursive* DNS nameservers were vulnerable to DNS cache poisoning.

Unfortunately, the decision to immediately push out a DNSSEC requirement for the Federal government was rushed and not sufficiently developed. As a result, six years later, DNSSEC remains unadopted by the vast majority of nameservers on the internet and *not a single web browser* validates DNSSEC queries.

Background on the Domain Name System (DNS)

In order for the internet to function efficiently, and allow users to navigate uniform resource locators (URLs) without having to use cumbersome numeric internet protocol addresses (IPs), the DNS system allows for human readable lookups (example.gov) that translate to IPs. DNS nameservers maintain these records and communicate to agents of users (e.g. a web browser or a command terminal, AKA a *user-agent* or *client*) upon request to translate a domain name URL to an IP.

There are multiple responses DNS nameservers may provide, for example : **authoritative answers, recursive querying**, and **cache querying**.

An **authoritative answer** from a nameserver indicates it is the nameserver which holds the record (known as the A record for IPv4 addresses, the AAAA record for IPv6 addresses) translating the URL into an IP.

Actual example from WhiteHouse.gov

```
~ dig whitehouse.gov
(...)

;; ANSWER SECTION:
whitehouse.gov.          20      IN      A       23.13.176.110
```

Recursive querying is used if the nameserver does not contain a record that maps a domain URL to an IP address. In this case, it *recursively* searches nameservers, starting with the root domain. Each nameserver in turn refers to additional nameservers until a record is found.

Example:

> "."[1] -> .gov -> whitehouse.gov -> subdomain.whitehouse.gov

Cache querying allows nameservers to store a DNS record without referring back to one of the authoritative nameservers. Depending on the network topology between the client and various nameservers, and the client's own DNS settings, there might be multiple caches of DNS records used to resolve a DNS request. Caches of DNS records are maintained by the "time-to-live" setting (TTL). This setting is a variable and is set by an authoritative nameserver. In the example above, the WhiteHouse.gov record has a TTL of 20 seconds.

Nameservers can implement any combination of the above actions. While only authoritative nameservers are technically required for the internet to function, given the current size and complexity of the internet, such a network would not be performant.

DNS Security (DNSSEC)

The problem with the DNS system as originally conceived is that there is no way to cryptographically validate the authenticity of any DNS packet[2]. Servers accept DNS packets before transport layer security (TLS/SSL) is negotiated. DNSSEC was conceived as the DNS analogue of what HTTPS brings to HTTP.

Before the Kaminsky vulnerability and mandated DNSSEC deployments, DNS servers traditionally listened on User Datagram[3] Protocol (UDP) port 53 for DNS packets with Transaction ID (TXID) responses that match its queries. Because the internet is a distributed network, any adversary with an advantageous topological position on the internet could intercept DNS packets, posit in those packets that malicious IPs *actually* match a domain URL, and then forward the DNS packet back to the nameservers (and eventually the client). This is the fundamental technique behind **DNS poisoning[4].**

Before August 2008, nameservers did not use sufficiently random TXIDs, allowing adversaries to potentially poison DNS by "guessing" enough TXIDs across the limited entropy or "randomness" of the TXID - there are only 16 bits in this field. To put this in easy to understand terms, it only requires on average 32, 768 attempts to correctly guess an ID of this length - well within the computational ability of any modern device.[5]

---

[1] The dot "." is the symbol for the root domain.
[2] A "network packet", the basic formatted unit of data transfer on the internet.
[3] Interchangeable with the definition of a packet.
[4] For an extensive analysis of the Kaminksy bug see: http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html
[5] http://tools.ietf.org/html/rfc5452

In order to increase the search, or "guess" space of TXIDs after the Kaminsky vulnerability became known, DNS providers started to **randomize their ports** along with **strengthening the randomness** of their TXIDs. Now, depending on the quality of the implementation by the vendor, the chance of a DNS poisoning attack might now be .00000001%, if not lower.

Since computational ability is always increasing, an even more secure solution was sought.[6] DNSSEC uses asymmetric cryptography in a public key infrastructure system to sign DNS records. A fulsome explanation of asymmetric cryptography is beyond the scope of this memo. Simply put, two keys capable of encoding information are created: a public key and a private key. By decoding a message encrypted with the private key by using the public key, a nameserver can gain certainty that DNS packets are being sent from a trusted source, or at least the certainty that the packets are sent from a source in possession of the private key in question.

In order for the DNSSEC system to provide value to an end-user client, each server in the chain must successfully and fully implement the DNSSEC standard. If **any link in the chain of trust is broken**, the security of the encryption (and thus the records it contains) **becomes completely useless**. The DNSSEC standard reflects this binary outcome. Current implementations create a significant gap in the chain, and distinguishes DNSSEC from other cryptographic design schemes, such as TLS/HTTPS.

The only thing communicated to the client is whether or not the Authenticated Data (AD) bit is set. An AD bit is only set if the entire chain of servers validates as DNSSEC[7]. Consequences of this design decision are far reaching. The RFCs themselves openly admit the fragility of the system. In RFC 3655:

> *"A resolver MUST NOT blindly trust the AD bit unless it communicates with a recursive nameserver over a secure transport mechanism..."*[8]

The RFC goes on to further clarify the need for "out of band" communications:

> *"The AD bit MUST only be trusted when the end consumer of the DNS data has confidence that the intermediary resolver setting the AD bit is trustworthy. This can only be accomplished via an out of band mechanism..."*

---

[6] Secure is relative in this context. Most cryptologists would not consider something secure until it takes centuries to crack. No one will spend centuries running a crack; but new mathematical techniques or poor implementations may suddenly downgrade the time to crack a particular cryptographic system. Starting with a timescale of centuries under “worst case scenarios” gives the system administrator some buffer.

At 18F, we build systems with an eye cryptographic security at the "global level". It should require an energy output equal to that necessary to boil all of the Earth’s oceans in order to crack any of our cryptographic systems on human timescales, given known algorithms..

This is not meant to be flippant. Innovations can suddenly render the inconceivable cheap and common. The SHA-1 algorithm will likely be able to be broken with only $700,000 in cloud computing costs and one year of running time in 2015. If an adversary is willing to pay more for dedicated GPUs instead of CPUs, the time to crack may be significantly reduced (ex: see discussion below of SHA-1). 18F constantly monitors the state of cryptography - since the main adversaries of 18F are primarily other nation-states,

[7] See https://www.ietf.org/rfc/rfc3655.txt and https://tools.ietf.org/html/rfc4035#section-3.2.3

[8] The rest of this paragraph concerns itself with message authentication alternatives in order to achieve trust, both of which are since deprecated since they rely on the insecure MD5 hashing algorithm.

RFC 3655 lists three acceptable out of band mechanisms to establish trustworthiness: fiat (pointless for the use case of the web), personal (similarly unworkable on the web), and specific knowledge of the DNSSEC setup. (semi-workable, but only for the most expert of technologists familiar with DNSSEC in general and cryptography in the specific - also introduces the problem of verifying the setup communicated is in fact the setup implemented).

While the above design does speak to potential advantages in a *corporate network* environment, especially where trusted system administrators directly control the configuration of all clients, it does not speak to the use case of the web (an untrusted network) - rather, it reveals several critical gaps to make DNSSEC practically unworkable on the web.

Specific DNSSEC Gaps

1. **No web browser has implemented a user interface indication that DNSSEC has failed or succeeded**

   The most common type of client used to access DNS is a web browser. No web browsers present any indication to the user that the AD bit is *not* set and thus DNSSEC has failed. Web browser users have no way of knowing whether or not their DNS lookup has been secured against poisoning. Assessing the AD bit directly, and the chain of encryption necessary used to set it, is a skill that requires notable technological capability. Many skilled developers don't know how to assess DNSSEC, or even know that it exists. DNSSEC succeeds or fails silently.

   A very small number of developers have lobbied for DNSSEC checks to be incorporated into web browsers, but none have been successful. An even *smaller* number of developers have attempted to create browser extensions that add DNSSEC capabilities. None of these extensions have been validated or audited by third-parties - mainly due to their unpopularity. As of September 2014, only 7,701 users on the entirety of the web can be confirmed to have downloaded these extensions.[9]

2. **Multiple gaps in the chain of trust**

   In order to preserve backwards-compatibility with DNS and to provide accurate guidance to DNSSEC enabled clients, DNSSEC nameservers refuse to set the AD bit if any portion of the chain is not DNSSEC compliant.

   Increasingly, Federal websites rely on multiple third parties to furnish content, or they use a content delivery network (CDN) like Akamai. No known CDN is entirely DNSSEC compliant and there are almost no known DNSSEC compliant third party content

[9] Only Chrome and Firefox publish this data - usage stats for Internet Explorer and Safari are unavailable. Since those extensions require additional technical knowledge to successfully install, the likelihood they are *more* popular than the "one-click" versions is implausible.

furnishers. In either case, but especially in the case of CDNs[10], any DNSSEC compliance is superficial and provides no real security to the end-user.

Additionally, many DNS providers and Certificate Authorities (CAs) are not DNSSEC compliant. The most popular vendors for these services in the Federal community, VeriSign[11] and Symantec[12], are not DNSSEC compliant themselves, *completely* undermining the chain of trust.

Lastly, if an end-user's internet service provider (ISP) sets up non-DNSSEC compliant recursive or caching nameservers between the end-user and a trusted DNSSEC compliant nameserver, the chain of trust breaks. Many ISPs do this and only a very small portion of end-users know how to properly set their own DNS servers.

3. **The AD bit can be forged**

Even if DNSSEC is successful, the AD bit is in the part of the packet that has no assurances whatsoever. Any adversary sitting between the client and the resolving nameserver can add their own records to a DNS packet and set the AD bit. Per IETC RFC 3655: "The AD bit MUST only be trusted when the end consumer of the DNS data has confidence that the intermediary resolver setting the AD bit is trustworthy."[13] DNSSEC does not defeat DNS poisoning or person in the middle attacks. It just moves the attack surface to a position that is easier to attack in the first place.[14] As the RFCs make clear, the AD bit can only be trusted if communicated via a secure channel. The only practical way of communicating this information over the web then becomes TLS, which cannot (currently) be used to negotiate an initial DNS lookup, thus creating an impossible chicken and egg scenario.

4. **SHA-1, the most common encryption method used with DNSSEC can be cracked**

Most of the Federal DNSSEC signatures evaluated by the author use the SHA-1 cryptographic hash function, an algorithm whose use for signature validation in TLS is in the process of being aggressively deprecated[15] NIST is also in concurrence and has formally *disallowed* SHA-1 since January 2014.[16] In 18F's analysis, any system using SHA-1 as part of its cryptography is without merit at best and dangerous at worst.

Any of these flaws taken individually means DNSSEC cannot be successfully implemented end-to-end. Taken together, they indicate a system that needs a serious re-evaluation.

---

[10] As an example. www.whitehouse.gov is just a CNAME (an alias) to Akamai infrastructure which is not signed from either the .gov or .net zones.
[11] http://dnssec-debugger.verisignlabs.com/www.verisigninc.com
[12] http://dnssec-debugger.verisignlabs.com/trustcenter.websecurity.symantec.com
[13] https://www.ietf.org/rfc/rfc3655.txt
[14] If an adversary controls the user's access point, no known system can completely verify that packets have not been modified.
[15] https://konklone.com/post/why-google-is-hurrying-the-web-to-kill-sha-1
[16] http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf

Current technical remediations

Until the flaws of DNSSEC can be remediated or a new system is implemented, **DNSSEC compliance is an expensive proposition that distracts from real security assurance**. 18F is committed to working with the internet community on implementing an end-to-end solution for DNS security, whether through DNSSEC or other solutions.

18F has already taken a critical remediation step by creating an internal policy that we are a "HTTPS Only" organization. a policy which has become the overarching policy of the US Government. Regardless of the data sensitivity our websites serve, we always establish an HTTPS connection using best in class TLS settings with our end-users. 18F believes *all* information the public exchanges with the Government should be private and secure. Without HTTPS, DNSSEC over the web is without merit or trust. With HTTPS, DNSSEC becomes an arcane and vestigial technological system.

All modern browsers have implemented excellent user interface warnings if HTTPS fails. **Users must consciously accept the risk and click through warnings multiple times**. Even if DNS poisoning or a person in the middle attack is successful, an end-user would be prevented from going to the malicious site (since TLS would fail) until they purposefully accepted the risk. Browser vendors have progressively increased the severity of the language they use in describing these risks to end-users and are constantly researching better designs.[17]

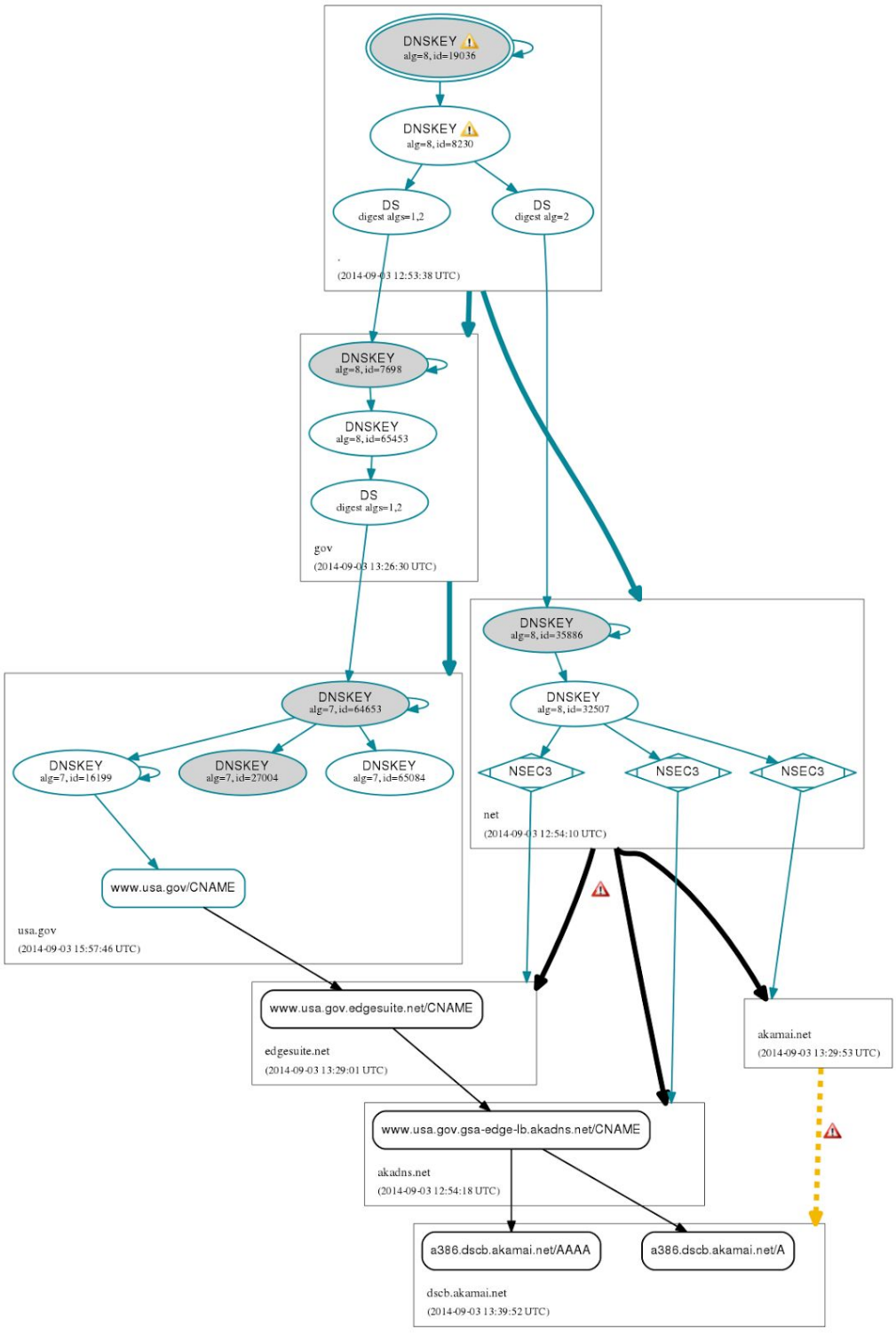
In addition, 18F's DNS provider is through Amazon Web Services (AWS) Route 53. These name servers have no recursive capabilities and thus are immune to currently known DNS poisoning methods. Since 18F, and the Federal government more broadly, cannot enforce end-to-end DNSSEC to any end-user, the responsibility for establishing a trusted connection for highly sensitive data exchanges remains on the end-user.

18F uses other technologies on the backend when exchanging sensitive data with Federal agencies, such as but not limited to the use of Virtual Private Networks (VPNs) or direct connections with Virtual Private Clouds (VPCs), or by simply avoiding DNS resolution altogether through the use of static IP addresses. **Ideally, none of the above systems are used.** Instead, agencies should focus on secure authentication and authorization combined with, at minimum, an out of band second factor.

Examples of current "DNSSEC" implementations:

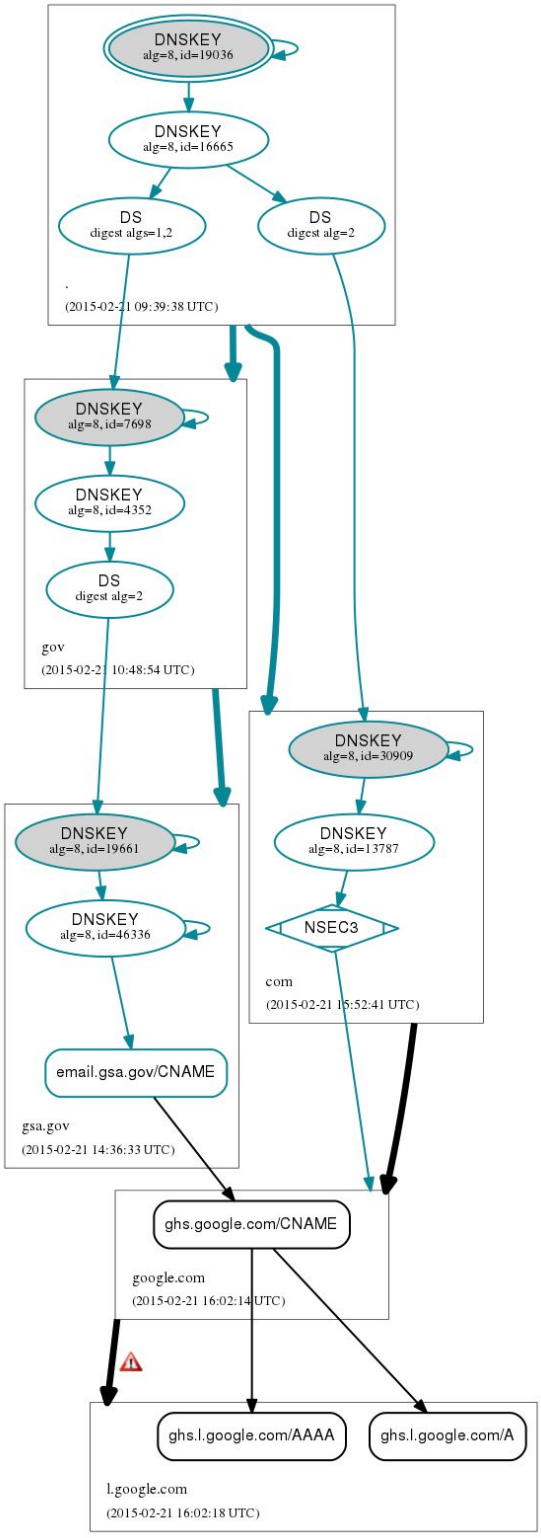
[17] http://www.adrienneporterfelt.com/chi-ssl-experiment.pdf

There are already multiple Federal DNSSEC implementations that exhibit the same traits as 18F systems. For example, here is the DNSSEC mapping (using DNSViz, a tool created by Sandia National Laboratories and Verisign) for www.usa.gov:

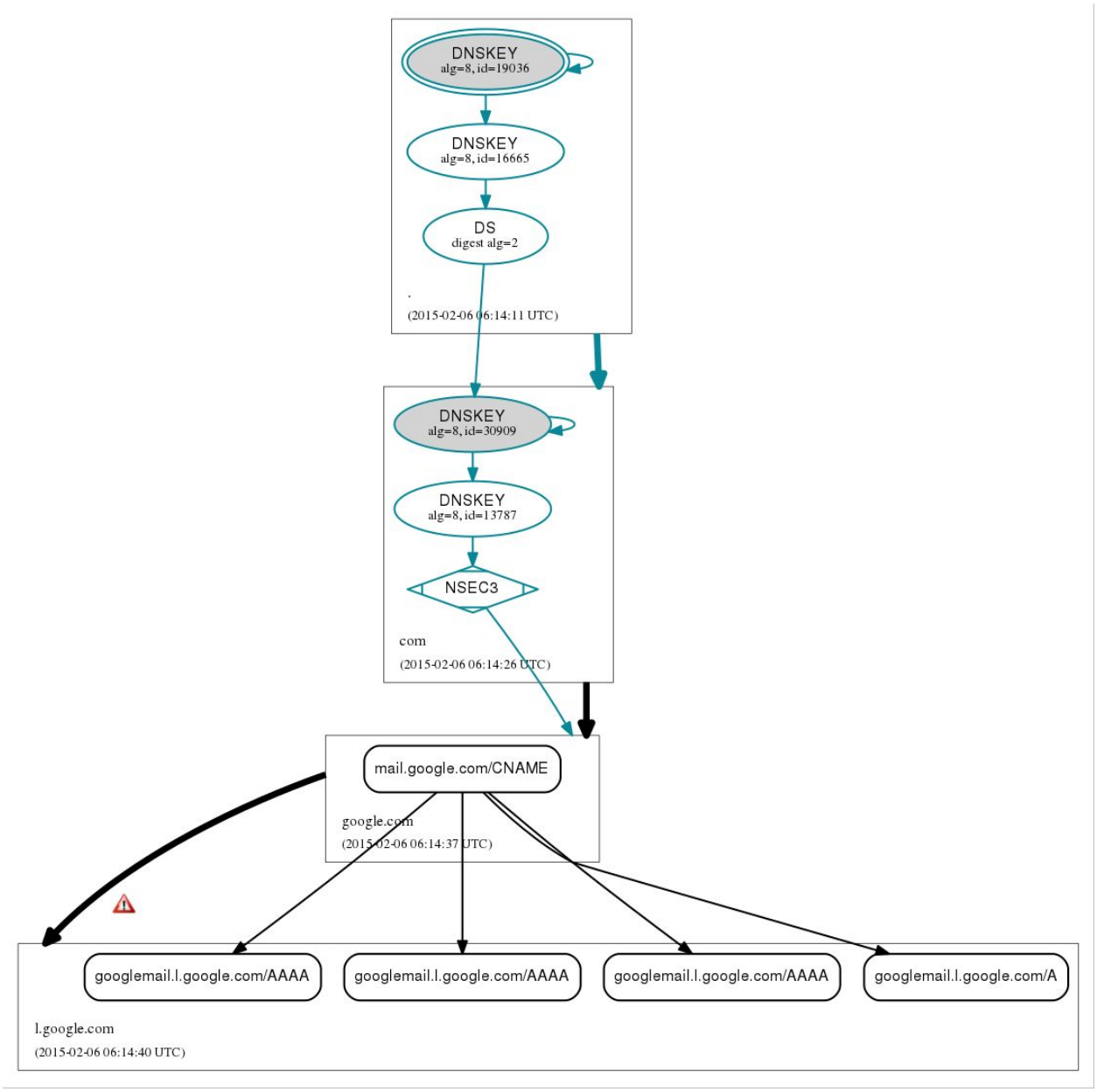


Note that the green lines represent DNSSEC compliant records, orange warning symbols indicate implementation issues, red warning signs indicate DNSSEC failures and black lines indicate no DNSSEC records at all. This system fails to implement DNSSEC between itself and its CDN, Akamai. This is very common. DNSSEC also fails for "backend" systems, like GSA

Google Apps email:



Even after GSA authentication is complete, the GSA email system still "fails" DNSSEC:


DNSKEY
alg=8, id=19036
DNSKEY
alg=8, id=16665
DS
digest alg=2
.
(2015-02-06 06:14:11 UTC)
DNSKEY
alg=8, id=30909
DNSKEY
alg=8, id=13787
NSEC3
com
(2015-02-06 06:14:26 UTC)
mail.google.com/CNAME
google.com
(2015-02-06 06:14:37 UTC)
googlemail.l.google.com/AAAA
googlemail.l.google.com/AAAA
googlemail.l.google.com/AAAA
googlemail.l.google.com/A
l.google.com
(2015-02-06 06:14:40 UTC)